Preliminary Information

Processor Family

BIOS Writers Guide

Version 0.8 of this document was released 5 October 1998.

INTRODUCTION

The IDT WinChip 2 is an x86-compatible CPU with unique features and restrictions. This guide should serve as a starting point for any BIOS programmers adapting their BIOS code to recognize and fully utilize the IDT WinChip 2. This is not a stand-alone document; it is meant as a companion to the IDT WinChip 2 Data Sheet. The Data Sheet provides the detailed information. This document also contains some (very few) tidbits of advance information about WinChip 3.

GENERAL COMPATIBILITY

The IDT WinChip 2 is highly compatible with standard ("586-class") x86 implementations. The IDT WinChip 2 also supports industry-compatible instruction set extensions that speed up certain graphics operations (AMD-3D Technology). The handling of SMI is compatible with the Intel Pentium (P54C). However, only integral clock speed multiples are supported. Currently the IDT WinChip 2 supports multipliers of 2, 3, 4 and 5x. The following table summarizes the IDT WinChip 2 usage of the BF pins as compared to the P54C. Please refer to the Data Sheet for more detailed information.

Bus Frequency Multipliers For WinChip 2

Pins			Clock Multiplier	
BF2	BF1	BF0	IDT WinChip 2	Intel P54C
1	0	0	reserved	2.5 x
1	0	1	3 x	3 x
1	1	0	2 x	2 x
1	1	1	4 x	1.5 x
0	0	0	reserved	
0	0	1	5 x	
0	1	0	4 x	
0	1	1	reserved	

The 100MHz bus frequency version supports a different set of clock multipliers, including fractional multipliers not supported by previous versions. These multipliers are summarized in the following table.

Bus Frequency Multipliers 100 MHz Bus

Pins			Clock Multiplier
BF2	BF1	BF0	IDT WinChip 2-100MHz Bus
1	0	0	2.5x
1	0	1	3 x
1	1	0	3.33 x
1	1	1	3.5 x
0	0	0	4.5 x
0	0	1	2.33 x
0	1	0	4 x
0	1	1	2.66 x

In this version (100 MHz Bus) of the WinChip 2 the multiplier information can be read from a machine-specific register (FCR4, MSR 0x10A) as follows:

Ratio	FCR4[5:2]	FCR4[1:0]
2.5X	0011b	00b
3.0X	0100b	00b
3.33X	1000b	01b
3.5X	0101b	00b
4.5X	0111b	00b
2.33X	0101b	01b
4.0X	0110b	00b
2.66X	0110b	01b

FCR4 Register Settings

Ratio = (FCR4[5:2] + 2) ÷ (FCR4[1:0] + 2)

The following code will calculate the appropriate ratio from the FCF4 register. The ratio is stored in the EAX register.

```
MOV     EDX, 00      // CLEAR EDX REGISTER
MOV     EAX, 00      // CLEAR EAX REGISTER
MOV     ECX, 10Ah    // READ MSR FCR4 (010Ah)
RDMSR                // RESULT GOES TO EDX AND EAX
MOV    EDX, EAX     // SAVE FCR[31:0] IN EDX
AND     EAX, 0003h   // CLEAR ALL BUT FCR4[1:0]
ADD     EAX, 0002h   // EAX = FCR4[1:0] + 2
MOV     EBX, EDX     // MOVE FCR[31:0] TO EBX
SHR     EBX, 2      // SHIFT RIGHT 2 PLACES
AND     EBX, 000Fh   // CLEAR ALL BUT FCR4[5:2] WHICH
                     // ARE IN LOW ORDER BITS IN EBX
ADD     EBX, 2      // EBX = FCR4[5:2] + 2
DIV     EAX, EBX     // EAX = RATIO
```

The Machine-Specific Registers (MSRs) are not currently compatible with a P54C except for the Timestamp Counter (more on that later). When an IDT WinChip 2 CPU is recognized, writes to and reads from MSRs should be avoided (with the exception of MSR 10h, the Timestamp Counter). There are some performance-enhancing features that can and should be enabled. Write-combining, weak read ordering and linear burst ordering are such features (described later).

The IDT WinChip 2 does not use a “P-rating” of any kind. Although “MMX-compatible” instructions are supported, the string “MMX” should not be displayed with the processor name. Furthermore, the “-S” string that Intel uses to identify CPUs that support SMI should not be used, even though SMI is supported. The only identifier that should be displayed is “IDT WinChip 2” and a megahertz rating. This megahertz rating is the actual internal clock frequency.

IDT WINCHIP 2 RECOGNITION

The first step is to recognize the IDT WinChip 2 and its revision level. Because each revision level can have unique features, both steps are necessary.

CPU recognition is accomplished through the use of the CPUID instruction. The CPUID instruction on the IDT WinChip 2 is compatible with that of other x86-compatibles, but with added functionality. Refer to the Data Sheet for more detailed information on CPUID. We will summarize only the features we use in recognizing and working with the IDT WinChip 2.

Operation of CPUID (Basic Functions):

Input value: eax: 0

Output values:

eax: 1

ebx: 0x746E6543

edx: 0x48727561

ecx: 0x736C7561

Note that this makes the vendor ID string "CentaurHauls"

Input value: eax: 1

Output values:

eax[3:0]	Stepping ID
eax[7:4]	Model
eax[11:8]	Family
eax[31:12]	Reserved
ebx	Reserved
ecx	Reserved
edx	Feature flags

The stepping ID, model, family and feature flags are defined compatibly. The IDT WinChip C6 returns an eax value of 0x54*z*, where *z* is the stepping number. The IDT WinChip 2 returns 0x58*z*.

Extended CPUID Functions

The extended functions are described in detail in the Data Sheet. We will describe only the one we use in detecting the CPU type and feature set.

The original IDT WinChip C6 supported one extended function.

Input value: eax: 0xC0000000

Output value:

eax: 0xC0000000

In other words, the input value must return unchanged. This indicates that only C6 features are supported.

The input value eax = 0x80000000 will also be returned unchanged for an IDT WinChip C6. This can be used instead of eax = 0xC0000000. For the IDT WinChip 2 and beyond, the input value eax = 0x80000000 produces the following output:

eax = 0x80000005 (highest extended cpuid function number)

ebx, ecx, edx reserved

The IDT WinChip 2 supports several extended functions in addition to the one described above. These should not be used unless it is previously determined that you are running on an IDT WinChip 2 per the above.

Input value: eax: 0x80000001

Pertinent output values:

eax[3:0] Stepping ID

eax[7:4] Model

eax[11:8] Family

eax[31:12] Reserved

edx[31] 1 if AMD-3D instructions supported

0 if AMD-3D instructions not supported

This information can be used to select an identification string to display at boot time. However, there is another architected means of determining the correct display string which might be simpler to implement, if that is all that is desired.

Input values: eax: 0x80000002 – 0x80000004 (three values)

Output:

Returns the name of the processor, suitable for BIOS to display on the screen (ASCII). The string can be up to 48 characters in length. If the string is shorter, the rightmost characters are padded with zero. The leftmost characters go in eax, then ebx, ecx, and edx. The leftmost character goes in least significant byte (little endian).

For example, the string “IDT WinChip 2” would be returned by extended function eax = 0x80000002 as follows:

eax = 0x20544449

ebx = 0x436E6957

ecx = 0x20706968

edx = 0x00000032

Since the string is only 13 bytes, the extended functions eax = 0x80000003 and eax = 0x80000004 return zero in eax, ebx, ecx and edx.

CLOCK FREQUENCY DETERMINATION

The best algorithms for determining the clock frequency make use of the RDTSC instruction. This instruction yields a running 64-bit count of the number of processor clocks since powerup. This count can be run against either of two independent clocks in the PC architecture. These two clock sources can be programmed to generate interrupts and the TSC count can be compared between two successive interrupts. This count is accurate enough to place the CPU into one of several “megahertz bins.” A system will generally support only several bus speeds, such as 50Mhz, 60Mhz, 66Mhz, etc. These frequencies,

multiplied by the available multipliers of 2, 3, 4 and 5, yield the aforementioned megahertz bins.

The two sources are the real-time clock, which will generate the so-called “periodic interrupt” and the timer tick, with its frequency of close to 18 times per second.

DTLOCK

DTLOCK prevents the table walk state machine from performing a read-modify-write sequence on updates to the Accessed and Dirty bits in the page directory/tables. DTLOCK is controlled by bit 8 of the Feature Control Register (MSR 0x107). This bit must be set to 1 for normal operation.

LINEAR BURST MODE

Linear burst mode improves performance in the cases where the first address of a burst read cycle is not a multiple of 16. Linear Burst Mode is enabled for the IDT WinChip 2 by setting bit 6 of the Feature Control Register (MSR 0x107) to 1. *This feature must be supported by the chipset in order to be used. Please refer to the individual chipset documentation for how to enable this feature.*

TRAIT MODE KEY

If system software is only concerned with programming the memory configuration registers, then it can read the MCR_CTRL register (MSR 0x120) and inspect the Trait Mode Key field (MCR_CTRL[19:17]). In the IDT WinChip 2 and later versions of the processor family the Trait Mode Key must be written to the Trait Mode control field (MCR_CTRL[8:6]) in order to activate the memory configuration registers.

In general system software can determine the processor version by comparing the Family and Model Identification fields returned by the CPUID standard function EAX=1.

If the processor version is not recognized then system software must not attempt to activate any machine-specific feature.

The following table indicates how to interpret the results of both methods.

Family	Model	Trait Mode Key MCR_CTRL[19:17]	Processor Version
5	4	0	IDT WinChip C6
5	8	1	IDT WinChip 2
5	9	1	IDT WinChip 3

WRITE-COMBINING AND WEAK READ ORDERING

These are features that are described in detail in the Data Sheet. Briefly, when write-combining is enabled, the IDT WinChip 2 will attempt to accumulate memory writes of less than a full dword. This reduces the memory access overhead and improves performance. This feature should be enabled when the IDT WinChip 2 (or the WinChip C6) is recognized. Weak read ordering allows read operations to be re-ordered ahead of writes to different cache lines, which can result in data being available to a program earlier.

The actual implementation of these features varies with the Feature Set Level, so it is imperative that this be checked before enabling the features.

These concepts are better described in the sample source code below. This is a very basic algorithm designed for simplicity to illustrate the concepts. Refinements to this algorithm can be made easily to accommodate other memory capacities. Please refer to the data sheet for more detailed information. Areas of improvement to this very basic algorithm might include better support for memory sizes not a power of two megabytes. The algorithm shown treats anything greater than the closest power of two megabytes as non-write-combining space, so for example if the memory size is 40 MB then the region from 32-40MB is not enabled for write-combining. Better support for the so-called “OS/2 memory hole” is also possible. When enabled, this hole is from 15-16MB. The sample algorithm will create a non-write-combining, non-weak-read-ordered hole from 12-16MB.

Note the use of cpuid with eax = 0xC0000000 to determine which IDT WinChip is installed. The sample code will also set the EDCTLB bit in the Feature Control Register, as required. Note also the use of the Trait Mode Key as described above.

Simple Sample Algorithm to Enable Write-Combining

```
; This sample source code is placed in the public domain.
; Centaur Technology, Inc. disclaims all warranties, express
; or implied,  and all liability, including consequential
; and other indirect damages, for the use of this source
; code, including the warranties of merchantability and
; fitness for a particular purpose.
; Centaur Technology, Inc. does not assume any
; responsibility for any errors which may appear in this
; program nor any responsibility to update it.



;=========================================================
; IDT WinChip 2 Write Combining Support
;=========================================================

; base/mask pairs for memory ranges

; 0 - 512K
base_0          equ     00000000000000000000000000000000b
mask_512K       equ     11111111111110000000000000000000b

; 512K 640K
base_512K       equ     00000000000010000000000000000000b
mask_640K       equ     11111111111111100000000000000000b
; 1M - 2M
base_1          equ     00000000000100000000000000000000b
mask_2          equ     11111111111100000000000000000000b

; 2M - 4M
base_2          equ     00000000001000000000000000000000b
mask_4          equ     11111111111000000000000000000000b

; 4M - 8M
base_4          equ     00000000010000000000000000000000b
mask_8          equ     11111111110000000000000000000000b
```

```
; 8M - 16M
base_8          equ     00000000100000000000000000000000b
mask_16         equ     11111111100000000000000000000000b
mask_12         equ     11111111110000000000000000000000b

; 16M - 32M
base_16         equ     00000001000000000000000000000000b
mask_32         equ     11111111000000000000000000000000b

; 32M - 64M
base_32         equ     00000010000000000000000000000000b
mask_64         equ     11111110000000000000000000000000b

; 64M - 128M
base_64         equ     00000100000000000000000000000000b
mask_128        equ     11111100000000000000000000000000b

; 128M - 256M
base_128        equ     00001000000000000000000000000000b
mask_256        equ     11111000000000000000000000000000b

; 256M - 512M
base_256        equ     00010000000000000000000000000000b
mask_512        equ     11110000000000000000000000000000b

; 512M - 1024M
base_512        equ     00100000000000000000000000000000b
mask_1024       equ     11100000000000000000000000000000b

; 1024M - 2048M
base_1024       equ     01000000000000000000000000000000b
mask_2048       equ     11000000000000000000000000000000b



; 2048M - 4096M
base_2048       equ     10000000000000000000000000000000b
mask_4096       equ     10000000000000000000000000000000b



; programming table for memory range registers

IDT_MCR_table   label   dword
                dd      base_2048
                dd      mask_4096

                dd      base_1024
                dd      mask_2048

                dd      base_512
                dd      mask_1024

                dd      base_256
                dd      mask_512

                 dd      base_128
```

```
                dd      mask_256

                dd      base_64
                dd      mask_128

                dd      base_32
                dd      mask_64

                dd      base_16
                dd      mask_32

                dd      base_8
                dd      mask_16

                dd      base_4
                dd      mask_8

                dd      base_2
                dd      mask_4

                dd      base_1
                dd      mask_2

                dd      base_512K
                dd      mask_640K

                dd      -1


MCRbaseno       equ     110h
MCR_CTRL        equ     120h

MCRvalue        equ     11111000000000000000001111b

BCRno           equ      145h
TLOCKbit        equ      3


;=======================================================
;Name     : Winchip Memory Features
;
; ** This routine should be called ONLY AFTER an IDT
;    WinChip CPU has been identified
;
;Function : To enable the IDT WinChip write combining and
;            weak read ordering
;
;Inputs:  edi = memory size in MB
;         bp = 1 if memory hole exists from 15 to 16MB
;              0 if not
;
;=======================================================


WinChip_Memory_Features  Proc    near

                pushad
```

```
; first assume C6
; see memory range descriptors for details

; bh has trait bits, bl = 0 if C6, 1 if WinChip 2
; traits=111b, bl=0
                mov     ebx, 11100000000b
; special IDT function
                mov     eax, 0C0000000h
                cpuid
                cmp     eax, 0C0000000h
                je      sub_1M     ; go ahead if C6

; now check feature level to see if we understand what to do
; we have to skip the programming if we are dealing with the
; wrong levels
                mov     ecx, MCR_CTRL
                rdmsr
                and     eax, 111b shl 17   ; check [19:17]
                cmp     eax, 1 shl 17  ; we only understand
                                       ;  a value of 001b
                jne     No_Memory_Features  ;skip if not 1


; here if WinChip 2
; traits to turn on are weak read ordering and byte
; combining

; traits=10001b, bl=1
                mov     ebx, 1000100000001b


sub_1M:
        ; always set up 0-512K region

                mov     ecx, MCRbaseno

                mov     edx, base_0
                mov     eax, mask_512K
                or      al, bh  ;OR in the local trait bits

                wrmsr

; now calculate offset into table to program the rest of the
;  memory range registers
; bsr returns the bit index of the highest 1 (largest power
;  of 2 in the number)

                bsr     ecx, edi
                neg     ecx
                lea     esi, [8*ecx+offset IDT_MCR_table+96]


; esi now contains the offset

                 mov     ecx, MCRbaseno+1
 program_one_mcr:
```

```
                mov     edx, cs:dword ptr [si]

                cmp     edx, -1
                je      short mcr_done

                mov     eax, cs:dword ptr [si+4]

                cmp     edx, base_8
                jne     short around_hole_check
                or      bp, bp
                jz      short around_hole_check
                mov     eax, mask_12    ; if hole enabled

around_hole_check:
                or      al, bh  ;OR in the local trait bits

                wrmsr

                add     si, 8
                inc     ecx
                cmp     ecx, MCRbaseno+8
                jb      program_one_mcr
mcr_done:

        ; finally set up MCR_CTRL
; no trait mode control for c2c
                mov     edx, 0
                mov     ecx, MCR_CTRL
                test    bl, bl    ; check for WinChip 2
                jz      c2cwc     ; if C6

; if WinChip 2, need to get trait mode control bits
                rdmsr       ; bits 19:17 trait mode control
; isolate these bits
                and     eax, 111b shl 17
                shr     eax, 11      ; move them to 8:6
                mov     edx, eax   ; save the isolated bits

c2cwc:
                mov     eax, MCRvalue  ; get control value
                or      eax, edx  ; OR in trait mode control
                mov     edx, 0    ; prepare to write back
                wrmsr

                mov     ecx, BCRno
                rdmsr
                or      ax, 1 shl TLOCKbit
                wrmsr

No_Memory_Features:
                popad
                ret

WinChip_Memory_Features  endp
```